Parallel Perceptrons and Training Set Selection for Imbalanced Classification Problems

Iván Cantador and José R. Dorronsoro *
Dpto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid, 28049 Madrid, Spain

ABSTRACT

Parallel perceptrons are a novel approach to the study of committee machines that allows, among other things, for a fast training with minimal communications between outputs and hidden units. Moreover, their training allows to naturally define margins for hidden unit activations. In this work we shall show how to use those margins to perform subsample selections over a given training set that reduce training complexity while enhancing classification accuracy and allowing for a balanced classifier performance when class sizes are greatly different.

1 INTRODUCTION

Most real world classification problems involve imbalanced samples, that is, samples where the number of patterns from one class (usually the most interesting one and that we term the positive samples) is much smaller than that from others. There are many examples of this situation (Cardie et al.; Dorronsoro et al.), as well as a large literature on this topic, for which many techniques have been applied. Basic examples are ROC curves (Weiss et al.), the alteration of the sample class distribution, either by oversampling the minority class (Breiman 84) or by undersampling either the majority class (Kubat et al.) or both classes (Chawla et al.). Moreover, sampling techniques are also in the core of the more sophisticated methods that arise from the well known boosting paradigm, where sample data can be either filtered (Domingo et al.) to achieve a similar proportion of easy and difficult examples, resampled (Freund) as to give a higher probability to difficult examples, or reweighted (Friedman et al.) so that difficult examples have a bigger weight on the error function.

In general, when dealing with imbalanced data sets, one may expect that some positive and negative patterns are easy to detect no matter what procedure is used for discriminant construction. Most of these patterns, termed "redundant" in (Kubat et al.), can be safely removed from the training set, as they will be nevertheless well represented in it by other patterns that we may consider "safe". On the other hand, some other patterns may be hopeless, in the sense that they cannot be correctly classified under any discriminant, as their features clearly set them in one class while they are labeled as belonging to other.

*With partial support of Spain's CICyT, TIC 01–572

It is thus convenient to exclude these "label noisy" patterns from training, as they are likely to muddle any classifier construction procedure. Finally, we are left with "borderline" patterns, which are difficult to handle as their classifier assignment may change because of small feature perturbations. They are nevertheless interesting for classifier training, as they are likely to define the class boundaries.

This suggests a general procedure in which redundant and label noisy patterns are iteratively taken out of the training set, while safe and bordeline patterns are kept. The problem is of course how to define the removal procedure. Many suggestions do appear in the literature (Breiman 98; Freund; Kubat et al.; Provost et al.), some of them revolving around the margin concept. In this work we shall propose a new procedure for training set reduction based on the concept of margin that arises naturally in parallel perceptron (PP) training (Auer et al.). Parallel perceptrons have the same structure of the well known committee machines (Nilsson), that is, they are made up of an odd number N of standard perceptrons P_i with ± 1 outputs, and the machine's one dimensional output is simply the sum of these perceptrons' outputs (that is, the overall perceptron vote count). They are thus well suited for 2–class discrimination problems, but it is shown in (Auer et al.) that they can also be used in regression problems, as they have indeed a universal approximation property. Another contribution of (Auer et al.) is to give a general and effective training procedure for PPs, in contrast with the mostly ad–hoc methods proposed before. A key part of this training procedure is a margin based output stabilization technique that tries to augment the distance of the activation of a perceptron from its decision hyperplane, so that small random changes on an input pattern do not cause its being assigned to another class. Although these margins are not defined on the one dimensional output of a PP and they have to be considered independently for each perceptron, they do provide a way to measure the relevance of individual patterns with respect the overall training set and to establish a pattern selection strategy. In fact, the margin value γ obtained after training reflects how far apart from the perceptron boundary hyperplanes are most of the training patterns and, hence, their influence on the training itself.

We shall briefly describe in section 2 the training of PPs, as well as their handling of margins, while in section 3 we will describe the overall training set relevant pattern selection procedure and shall also see how margins can be used to discard both redundant and label noisy patterns while allowing to retain those patterns most interesting for training purposes. In section 4 we will illustrate numerically the results provided by the pattern selection algorithm over 3 example databases obtained fom the UCI repository. As we shall see, in all of them we arrive at much smaller training subsets that nevertheless allow the construction of effective PP classifiers.

2 PARALLEL PERCEPTRON TRAINING

Parallel perceptrons can be used both for regression and classification problems. To simplify things up, we shall briefly describe next their use for classification (the one we are interested in); see (Auer et al.) for a more complete description of general PP training. The transfer function of a PP has a very simple structure. Assume we are working with H perceptrons,

each with a weight vector W_i and a threshold h_i. For a given input X, the output of perceptron i is then $P_i(X) = s(W_i \cdot X - h_i) = s(act_i(X))$, where $s(\cdot)$ denotes the sign function and $act_i(X) = W_i \cdot X - h_i$ is the activation due to X of perceptron i. In other words, $P_i(X) = 1$ iff $W_i \cdot X > h_i$. Then, the PP output with respect to X is

$$f(X) \quad = \quad \sum_{1}^{H} P_i(X) = N_+(X) - N_-(X),$$

where $N_+(X)$ and $N_-(X)$ denote respectively the number of perceptrons having either $+1$ o -1 as their output. We will assume that each input X has an associated ± 1 label l_X; it is then clear that X has been correctly classified if $l_X f(X) > 0$. Now, if a pattern X such that, say, $l_X = 1$, has not been correctly classified, it turns out that $N_+(X) < N_-(X)$ and one should try to decrease $N_-(X)$, for which Rosenblatt's rule applied to those P_i such that $P_i(X) < 0$, i.e.,

$$W_i := W_i + \eta X, \quad h_i := h_i + \eta,$$

is a natural option. The same update equations can be used when $l_X = -1$ just changing the $+$ sign to a $-$ and applying them to those P_i such that $P_i(X) > 0$. Both update procedures can thus be written in a unified form as

$$W_i := W_i + \eta l_X X, \quad h_i := h_i + \eta l_X, \tag{1}$$

with the update rule being applied to those P_i such that $l_X P_i(X) < 0$. Weight updates can be done either on–line or in batch mode; here we shall just consider batch PP training.

The second component of PP training is a margin–based output stabilization procedure. Notice that if $W_i \cdot X \simeq h_i$, small changes on X may cause a wrong class assignment for a small perturbation of X. To avoid this instability, the p–delta rule also applies the update (1) when a pattern X is correctly classified but still

$$0 < l_X act_i(X) < \gamma.$$

The value of the margin γ is also adjusted dynamically from a starting value. A simple way to do so is to increase it for a given X if $l_X act_i(X) \geq \gamma$ for all perceptrons and to decrease it if this fails for at least one perceptron. Values proposed in (Auer et al.) for this are a moderate increase $\gamma := \gamma + 0.25 \times \eta$, but a stronger decrease $\gamma := \gamma - 0.75 \times \eta$, where η is some appropriately chosen margin learning rate. Notice that for the margin to be meaningful, weights have to be normalized; we will simply do so after each batch pass.

In spite of their very simple structure, PPs do have a universal approximation property. Moreover, as shown in (Auer et al.), PPs provide results in classification and regression problems as good as those offered by procedures such as MLPs and C4.5 decision trees. Finally, their training is very fast as the update (1) is much simpler than standard backpropagation, for the training information required by a perceptron is just one bit, depending on whether a pattern has been correctly classified or not, in contrast with the gradient–based generalized error information required by MLPs. Moreover, if we omit for simplicity updates due to

margins, the overall training complexity for a PP is $O(N_W DH)$, with N_W the number of patterns incorrectly classified, compared with the $O(NDH)$ complexity of MLPs, with N the full number of patterns; as training advances, we should have $N_W \ll N$.

3 Training pattern selection

The final margin value γ obtained when PP training ends reflects how far apart from the perceptron boundary hyperplanes are most of the training patterns. It is thus natural to establish for each perceptron 4 main "zones", highly incorrect, borderline incorrect, borderline correct and highly correct, according to whether a pattern's normalized activation, that is, the product $l_X act_i(X)$, with l_X the ± 1 label of X and $act_i(X)$ the activation of X over the i–th perceptron, is in the intervals $(-\infty, -\gamma]$, $(-\gamma, 0]$, $(0, \gamma]$ and (γ, ∞) respectively. With a slight abuse of the language, we shall also call a pattern's normalized activation its "margin". These zones clearly correspond to the label noisy, borderline and redundant cathegories mentioned before. It would be tempting to term those vectors close to γ as "support vectors" but observe that a concrete pattern may provide a clear positive γ margin on a perceptron while having a wrong negative γ margin over another.

We will define our pattern selection procedure using the margin values obtained after PP training ends. To do so, we will denote as $\mathcal{T}_R$ the set of redundant patterns and by $\mathcal{T}_N$ the set of label noisy patterns. Recall that $X \in \mathcal{T}_R$ if all N perceptrons provide a correct margin $\geq \gamma$ for it and that $X \in \mathcal{T}_N$ if all N perceptrons show a wrong margin $< -\gamma$ on its activations. After each training iteration i we shall remove the $\mathcal{T}_R^i$ and $\mathcal{T}_N^i$ subsets from the initial training set $\mathcal{T}^i$ of the previous iteration, arriving to the new training set $\mathcal{T}^{i+1}$. The iterations proceed while the training set g and acc^+ accuracy factors (see below) do not decrease.

Notice that on each iteration we keep borderline paterns, that is, those patterns X for which at least one perceptron gives a pattern margin between $-\gamma$ and γ. In fact, we may keep patterns for which all perceptrons give a wrong negative margin between $(-\gamma, 0)$; that is, all perceptrons may classify it wrongly, although with a not too high margin. We may expect thus that such a pattern may be correctly classified in a subsequent iteration. It is natural to expect that these borderline cases correspond to difficult but still learnable patterns. In fact, most training patterns do belong to this cathegory, so subsequent PP training somehow concentrates on these hard patterns; in view of the well known results on boosting, this is a natural way to proceed. However, we shall also discard the probably still abundant remaining borderline negatives, that we shall denote as $\mathcal{T}_{B-}$, because we want the minority class to have more relevance during the training process; thus, the PP is built using only all the borderline positive patterns and the borderline correct negative patterns. The last PP is then used over the test set to determine the reported values of the overall accuracy acc_{Te} and a value g_{Te} (see below) that measures how well balanced is this accuracy among positive and negative classes. The pseudocode of the general procedure is thus:

```
trSetReduction(Training_set Tr, Test_set Te)
  g_Tr = 0;
  acc+_Tr = 0;
  trainPP(Tr, g, acc+, W, gamma);        // Update PP weigths W and margin

  while(g >= g_Tr and acc+ >= acc+_Tr)   // Reduce Tr while g, acc+ improve
    W_PP = W;                            // W_PP: weights of best PP so far
    g_Tr = g;
    acc+_Tr = acc+;
    find(Tr, Tr_R, Tr_N, Tr_B-, gamma);  // Remove redundant, label noisy
    remove(Tr, Tr_R, Tr_N, Tr_B-);       // and all-wrong negative patterns
    trainPP(Tr, g, acc+, W, gamma);

  calcAccG(Te, W_PP, acc_Te, acc+_Te, acc-_Te, g_Te);
```

We finally point out that although it may be seem sensible at first sight, we will not discard borderline correct patterns $\mathcal{T}_{B+}$ with small margin. Given the characteristics of the data set used, such a procedure has in some cases the danger of removing all positive patterns, thus leaving us with a useless final PP that classifies all test patterns as negative.

4 NUMERICAL RESULTS

Simple accuracy, that is the percentage of correctly classified patterns, is not a relevant criterium in imbalanced class problems, as it would be fulfilled by the simple (and very uninteresting) procedure of assigning all patterns to the (possibly much larger) negative classes. Other criteria are thus needed, usually defined in terms of the elements of the well known confusion matrix, as defined in table 1, such as precision (the ratio $A/(A+C)$) or recall (the ratio $A/(A+B)$), ROC curves, or other. Here we shall use a measure first proposed in (Swets), the value of g defined as

$$g = \sqrt{a^+ a^-},$$

where $a^+ = A/(A+B)$ and $a^- = D/(C+D)$ are the partial accuracies over the positive and negative classes. The goal g attains is to measure the balance of the positive and negative class accuracies.

	guessed +	guessed −
true +	A	B
true −	C	D

Table 1: Confusion matrix for two class problems.

Problem set	initial g	final g	initial Tr set	final Tr set	ave. # iters
glass	90.5	94.0	184	119	0.63
vehicle	64.1	72.4	635	97	2.13
vowel	88.5	91.5	792	95	2.28

Table 2: Learning results for parallel perceptrons. The table shows a balance between the accuracy over the positive and negative classes, a large reduction of the training sets and a general improvement of g values after this reduction.

We shall use 3 problem sets from the UCI database, the glass, vowel and vehicle sets, that provide well known examples of highly imbalanced positive and negative patterns, that make classifier construction quite difficult, as discriminants may tend to favor the (much) larger negative patterns over the less frequently positive ones. All are multiclass problems but, as done in (Kubat et al.) we shall consider them as 2–class problems, with a small interesting positive pattern set and a larger negative one. The positive class will be the seventh class for the glass problem, the class 0 in the vowel problem and class 1 in the vehicle problem.

PP training has been carried out as a batch procedure. In all examples we have used 3 perceptrons and starting parameters $\gamma = 0.05, \eta = 10^{-2}$. The η learning rate has been used for both weight and margin updates; it does not change if the training error diminishes, but is changed to 0.9η if it augments. Training epochs have been 250 in all cases; thus the training error evolution has not been taken into account to stop the training procedure; anyway, it has an overall decreasing behavior. In all cases we have used 10–times K–fold cross validation, using all instances to compute the average values reported. More precisely, the overall data set has been randomly split in K subsets and on each training stage, $K-1$ of which are combined on a $\mathcal{S}_{tr}$ set used for training purposes. The final PPs' behavior has been computed on the remaining, unused subset, that we keep for testing purposes.

Table 2 presents average values of the cross–validation procedure just described. As it can be seen, g values are larger after training set reduction, markedly so in the vehicle case. Moreover, training set reduction is quite considerable, going from a reduction rate of 1.54 for the glass data set to a quite large 8.33 value for the vowel data set. Notice also that the number of pruning iterations is quite small. Therefore, training set reduction is achieved quite fast, to which the simple structure of PPs and their efficient training also help.

For comparison purposes, table 3 reports the PP g values, those obtained using a standard multilayer perceptron (for which no training set reduction is performed), and the results reported in (Kubat et al.) and obtained applying another training set reduction method for Quinlan's C4.5 decision trees and a 1–nearest neighborhood discriminant. In this latter case, we observe that although the cross–validation parameters used here are the same that in (Kubat et al.), the training sets actually used are different, so those values have to be compared with some care with those reported here. It can be seen in the table that g values for MLPs and PPs are quite similar, except for the vowel dataset, where MLPs give the best g value overall. Moreover, compared with the results in (Kubat et al.), PPs give better g

Problem set	K	PP	MLP	1–NN	C4.5
glass	7	94.0	93.4	96.6	84.5
vehicle	4	72.4	74.9	66.8	69.4
vowel	5	91.5	96.9	90.9	84.0

Table 3: g values for 3 datasets and different classifier construction procedures. It can be seen that parallel perceptrons give best results in the glass data set, while an MLP gives markedly better results in the vowel problem. The table also reports the number K of subsets used in the cross validation process.

values than C4.5 and 1–NN for the vehicle and vowel dataset, while for the glass dataset, its g value is slightly larger than that of 1–NN but much better than that of C4.5. As a summary of these results we point out that PPs' performance is comparable to that obtained with MLPs, that nevertheless have a more complex structure and a costlier training, while it improves on that of 1–NN and more on that of C4.5.

5 Conclusions and further work

In this paper we have proposed a new procedure for training set reduction based on the concept of margin that arises naturally in parallel perceptron training. Although different from margins obtained in, say, support vector machines, they also contribute to the stabilization of PP outputs and, hence, may improve on PPs' generalization abilities. This has been verified in the 2–class problems studied here. They are representative of imbalanced class problems, where the discrimination of a small positive class may be damaged by the much larger number of negative samples. We have shown that PPs can be used in a natural way to balance the number of positive and negative samples while ensuring a good generalization, as shown in the test set g values reported here, that demonstrate not only a good overall test set accuracy, but also that it is well balanced among positive and negative classes.

This property, together with the very fast training of PP, may make them quite useful on large dimension imbalanced problems, an area of considerable interest as many interesting problems (text mining, microarray discrimination) belong to it. This and other questions, such as PP use in active training, and improvements in their performance, either by combining PPs through boosting or enlarging their parameter space, are under study.

References

P. Auer, H. Burgsteiner and W. Maass, *Reducing Communication for Distributed Learning in Neural Networks*, Proceedings of ICANN'2002,Lecture Notes in Computer Science 2415 (2002), 123–128.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, **Classification and Regression Trees**, Wadsworth, 1983.

L. Breiman, *Arcing classifiers*, The Annals of Statistics 26 (1998), 801–849.

C. Cardie and N. Howe, *Improving Minority Class Prediction Using Case-Specific Feature Weights*, Proceedings of the Fourteenth International Conference on Machine Learning, D. Fisher (ed.), Morgan Kaufmann, 1997, 57–65.

N. Chawla, K. Bowyer, L. Hall and W. Kegelmeyer, *SMOTE: Synthetic Minority Oversampling Technique*, Journal of Artificial Intelligence Research 16 (2002), 321–357.

C. Domingo and O. Watanabe, *MadaBoost: A Modification of AdaBoost*, Proc. 13th Annu. Conference on Comput. Learning Theory, 180–189, Morgan Kaufmann, San Francisco, 2000.

J. Dorronsoro, F. Ginel, C. Sánchez, C. Santa Cruz, *Neural Fraud Detection in Credit Card Operations*, IEEE Transactions on Neural Networks, 8 (1997), 827-834.

T. Fawcett and F. Provost, *Adaptive Fraud Detection*, Journal of Data Mining and Knowledge Discovery 1 (1997), 291–316.

J. H. Friedman, T. Hastie and R. Tibshirani, *Additive Logistic Regression: a Statistical View of Boosting*, Dept. of Statistics, Stanford University Technical Report, 1998.

Y. Freund *Boosting a weak learning algorithm by majority*, Information and Computation 121 (1995), 256–285.

M. Kubat, S. Matwin, *Addressing the Curse of Imbalanced Training Sets: One-Sided Selection*, Proceedings of the 14th International Conference on Machine Learning, ICML'97 (pp. 179-186), Nashville, TN, U.S.A.

N. Nilsson, **The Mathematical Foundations of Learning Machines**, Morgan Kaufmann, 1990.

F. Provost and T. Fawcett, *Robust Classification for Imprecise Environments*, Machine Learning 42 (2001), 203–231.

A. J. Smola and B. Schölkopf, **Learning with Kernels**, MIT Press, 2002.

J. A. Swets, *Measuring the accuracy of diagnostic systems*, Science 240 (1998), 1285–1293.

G. M. Weiss and F. Provost, *The effect of class distribution on classifier learning*, Technical Report ML-TR 43, Department of Computer Science, Rutgers University, 2001.